Performance

Hung-Wei Tseng
Instruction set architecture

- Defines the language that your processor should speak
- Provides an abstraction for software to access hardware functions
  - Arithmetic operations: add, sub, mul, div
  - Logical operations: and, or, not
  - Memory access: load, store
  - Control: branch, jump
- Your software is a combination of these instructions
# MIPS v.s. x86

<table>
<thead>
<tr>
<th></th>
<th>MIPS</th>
<th>x86</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISA type</td>
<td>RISC</td>
<td>CISC</td>
</tr>
<tr>
<td>instruction width</td>
<td>32 bits</td>
<td>1 ~ 17 bytes</td>
</tr>
<tr>
<td>code size</td>
<td>larger</td>
<td>smaller</td>
</tr>
<tr>
<td>registers</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>addressing modes</td>
<td>reg+offset</td>
<td>base+offset, base+index, scaled+index, scaled+index+offset</td>
</tr>
<tr>
<td>hardware</td>
<td>simple</td>
<td>complex</td>
</tr>
</tbody>
</table>
Announcement

• Homework #1 due next Tuesday
  • Check google calendar http://goo.gl/JJyrXp
• Reading quizzes due next Wednesday
• Harsha’s new office hour on Fridays
  • Better assist you in solving homework problems
• Discussion session
  • Address on homework problems
  • Group office hours between TAs and you
Outline

• What is performance?
• What is the performance equation?
• What affects performance
Performance!
What do you want in a computer?

- Frame rate
- Responsiveness
- Real-time
- Throughput
- Cost
- Volume
- Weight
- Battery life
- Low power/low temperature
- Reliability
- Latency/Execution time

The most direct measurement of performance
Execution Time

- The simplest kind of performance
- Shorter execution time means better performance
- Usually measured in seconds

How long is it take to execution each of these?

Instruction Count!

How many of these?

Cycles per instruction * cycle time
Performance equation!
Performance Equation

Execution Time = \( \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}} \)

- How many instruction are executed?
- How long is it take to execute each instruction

- \( ET = IC \times CPI \times CT \)
- IC (Instruction Count)
- CPI (Cycles Per Instruction)
- CT (Seconds Per Cycle)
- 1 Hz = 1 second per cycle; 1 GHz = 1 ns per cycle
Performance Example

- Assume that we have an application composed with a total of 500000 instructions, the average CPI of the application is 2 cycle. If the processor runs at 1GHz, how long is it take to execution the application?

A. 500000 ns
B. 1000000 ns
C. 1750000 ns
D. 3500000 ns
E. None of the above

\[
\text{Execution Time} = \left( \frac{\text{Instructions}}{\text{Program}} \right) \times \left( \frac{\text{Cycles}}{\text{Instruction}} \right) \times \left( \frac{\text{Seconds}}{\text{Cycle}} \right)
\]

\[500000 \times 2 \times 1 \text{ ns} = 1000000 \text{ ns}\]
Performance Example

- Assume that we have an application composed with a total of 500000 instructions, in which 20% of them are the load/store instructions with an average CPI of 6 cycles, and the rest instructions are integer instructions with average CPI of 1 cycle. If the processor runs at 1GHz, how long is the execution time?

A. 500000 ns
B. 1000000 ns
C. 1750000 ns
D. 3500000 ns
E. None of the above

Execution Time = \[ \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}} \]

\[ 500000 \times (0.8 \times 1 + 0.2 \times 6) \times 1 \text{ ns} = 1000000 \text{ ns} \]
Speedup

• Compare the relative performance of the baseline system and the improved system

• Definition

\[
\text{Speedup} = \frac{\text{Execution time}_{\text{baseline}}}{\text{Execution time}_{\text{improved system}}}
\]
Performance Example

- Assume that we have an application composed with a total of 500000 instructions, in which 20% of them are the load/store instructions with an average CPI of 6 cycles, and the rest instructions are integer instructions with average CPI of 1 cycle.
- If we double the clock rate to 2GHz without improving the memory latency, the average CPI for load/store instruction will also become 12 cycles. What’s the performance improvement after this change?
  A. No change
  B. 1.25
  C. 1.5
  D. 2
  E. None of the above
Performance Example

• Assume that we have an application composed with a total of 500000 instructions, in which 20% of them are the load/store instructions with an average CPI of 6 cycles, and the rest instructions are integer instructions with average CPI of 1 cycle.

• If we double the clock rate to 2GHz without improving the memory latency, the average CPI for load/store instruction will also become 12 cycles. What’s the performance improvement after this change?

\[
\text{Execution Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}
\]

\[
ET_{\text{new}} = 500000 \times (0.8\times1 + 0.2\times12) \times 0.5 \text{ ns} = 800000 \text{ ns}
\]

\[
\text{Speedup} = \frac{ET_{\text{old}}}{ET_{\text{new}}} = \frac{1000000}{800000} = 1.25
\]
Identify the performance bottleneck

Why does an Intel Core i7 @ 3.5 GHz usually perform better than an Intel Core i5 @ 3.5 GHz or AMD FX-8350@4GHz?

A. Because the instruction count of the program are different
B. Because the clock rate of AMD FX is higher
C. Because the CPI of Core i7 is better
D. Because the clock rate of AMD FX is higher and CPI of Core i7 is better
E. None of the above

Sysbench 2014 from http://www.anandtech.com/
Identify the performance bottleneck

ET = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}

ET = \text{IC} \times \text{CPI} \times \text{Cycle Time}

Why does an Intel Core i7 @ 3.5 GHz usually perform better than an Intel Core i5 @ 3.5 GHz or AMD FX-8350@4GHz?

Every time when the question ask you about the “performance”, thinking about the performance equation first!
What affects performance
What affects performance?

- How many of the following factors affect performance?
  - Compiler
  - Instruction Set
  - Processor architecture
  - Instruction Set Architecture
  - Programmer
  - Process technology
  - Application
  - Programming languages

A. 3
B. 4
C. 5
D. 6
E. 7
How programmer affects performance?

- $ET = IC \times CPI \times CT$
- What can a programmer affect?
  - A. IC
  - B. IC & CPI
  - **C. IC, CPI & CT**
  - D. IC & CT
Demo: programmer & performance

- Row-major, column major
- Let’s identify where the performance gain is from!
  - Using “performance counters”
  - You may use “perf stat” in linux
  - You can also create your own functions to obtain counter values
How compiler affects performance?

• ET = IC * CPI * CT

• What can a compiler affect?
  A. IC
  B. IC & CPI
  C. IC, CPI & CT
  D. IC & CT
Demo: compiler & performance

- Compiler optimization can help reducing the instruction count
- Compiler optimization can improve CPI
  - Wise selection of instruction combinations
  - Use registers to eliminate loads and stores
Demo: storage and performance

• Machine implementation affects performance
• The same CPU, but different storage devices
  • Loading data from a H.D.D: access time ~ 10ms
  • Loading data from an S.S.D: access time ~ 100 us
Summary: Performance Equation

Execution Time = \( \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}} \)

- \( \text{ET} = \text{IC} \times \text{CPI} \times \text{Cycle Time} \)
- \( \text{IC} \) (Instruction Count)
  - ISA, Compiler, algorithm, programming language
- \( \text{CPI} \) (Cycles Per Instruction)
  - Machine Implementation, microarchitecture, compiler, application, algorithm, programming language
- \( \text{Cycle Time} \) (Seconds Per Cycle)
  - Process Technology, microarchitecture, programmer